REMARKS

Claims 1, 2, 4 - 8, 11, 12, 14 - 22 and 24 are pending. By this amendment, claim 11 is amended. Reconsideration and issuance of a Notice of Allowance are respectfully requested.

REJECTIONS UNDER 35 U.S.C. § 112

On page 2 the Office Action rejects claims 11, 12, and 14-17 under 35 U.S.C. § 112, second paragraph because of insufficient antecedent basis for one term in claim 11. This rejection is respectfully traversed.

Claim 11 is amended to correct antecedent basis of its recited elements. Accordingly, claim 11 is patentable under of 35 U.S.C. § 112, second paragraph. Withdrawal of the rejection of claims 11, 12 and 14 - 17 under 35 U.S.C. § 112, second paragraph is respectfully requested.

REJECTIONS UNDER 35 U.S.C. § 103(a)

On page 3 the Office Action rejects claims 1, 2, 4, 5, 6, 8, 11, 12, 14, 17-22 and 24 under 35 U.S.C. § 103(a) over "Systems Management: Application Response Measurement (ARM) API" (hereafter ARM API) in view of U.S. Patent 6,633,908 to Leymann, et al. (hereafter Leymann). This rejection is respectfully traversed.

Claim 1

In rejecting the claim 1, the Office Action states that ARM API teaches all that is recited except that ARM API does not "explicitly recite the translation of the performance information into a generic output." However, the Office Action then asserts that Leymann "teaches these features" and that it would have been obvious to combine Leymann and ARM API to arrive at the invention recited in claim 1.

ARM API is directed to a method for measuring service levels of applications in which a number of performance measurement agents collect response time information. See ARM API pages 16 - 17. To enable the agents to collect this information, each application must first be modified to include code that invokes the various performance functions. See ARM API page 5. ARM API does not disclose or suggest use of a generic output.

Similar to ARM API, Leymann is directed to performance monitoring of applications. However, Leymann departs from ARM API by not requiring modification to each application so as to invoke system calls. Instead of instrumenting applications, Leymann instruments invocation agents 202 and uses these invocation agents to collect the desired performance information. See column 8, lines 1 - 23. Leymann does not disclose or suggest use of a generic output.

Applicants assert that Leymann and ARM API cannot be combined because Leymann is distinct from ARM API in at least one important aspect, as noted above: ARM API requires invasive instrumentation of an application to be monitored. Leymann avoids this onerous requirement by "instrumenting the [monitoring] agent instead." See col. 7, line 67 - col. 8, line 1. Leymann's solution "provides application response measurements without any modification of the application being measured. As a consequence [and in contrast to ARM API] no special code has to be added to new or existing applications to enable them for response measurement." See col. 8, lines 3 - 7. Moreover, Leymann's invocation [measurement] agent 202 is a generic component so that instrumentation of the measurement agent has to take place only once. Thus, Leymann uses a generic <u>agent</u> to collect performance metrics from any number of applications. In contrast to Leymann, ARM API separately instruments each application to collect performance metrics. Accordingly, the combination of Leymann and ARM API would result in a non-working device or method.

In contrast to ARM API and Leymann, the invention recited in claim 1 says nothing about instrumenting applications or instrumenting monitoring agents. claim 1 does not recite a generic monitoring agent as in Leymann. Claim 1 does not recite instrumenting applications as in ARM API. Claim 1 puts no restrictions on the monitoring agent as in Leymann. Instead, and in contrast to Leymann and ARM API, the invention recited in claim 1 is a method for dynamically determining the health of a service including the step of "translating the collected service performance information into a generic output relating to current operational performance of the service." Leymann uses a generic monitoring agent. claim 1 recites translating the collected performance information into a generic output.

More specifically, claim 1 recites a method for determining the service health of a computer system. The method includes gathering performance information and translating the gathered performance information, into a "generic output." The "generic output" is an abstracted set of consistent service health metrics that can be provided to the performance monitoring tools such that the performance monitoring tools may use the health metrics without needing to know how the health metrics were derived. This method decouples the performance tool implementation from the metric derivation and removes dependencies between the services, their implementation, and the management tool set. For example, a tool may use the generated service level violation metric to generate an alert when violations raise above a threshold. The performance monitoring tools do not need to know anything about the service, its instrumentation, or how the service level metric is calculated. The tool simply compares the resultant metric against its threshold. The performance monitoring tool uses a programmatic interface library call or script interface to access health metrics for all current services. If the underlying application changes, the current version of the performance monitoring tool is unaffected because of this consistent interface. As a result, the system administrator does not necessarily need to install a new version of the performance monitoring tool.

The feature of the "generic output" is defined in the specification, for example, at page 5, lines 3 - 20:

> However the performance information is gathered, the apparatus and method translate the gathered performance information, or metrics, into health metrics. The result is an abstracted set of consistent service health metrics that can be provided to the performance monitoring tools such that the tools may use these metrics without needing to know how the health metrics were derived. This decouples the performance tool implementation from the metric derivation and removes dependencies between the services, their implementation, and the management tool set. For example, a tool may use the generated service level violation metric to generate an alert when violations raise above a threshold. The performance monitoring tools do not need to know anything about the service, its instrumentation, or how the service level metric is calculated. The tool simply compares the resultant metric against its threshold. The performance monitoring tool uses a programmatic interface library call or script interface to access health metrics for all current services. If the underlying application changes, the current version of the performance monitoring tool is unaffected because of this consistent interface. As a result, the system administrator does not necessarily need to install a new version of the performance monitoring tool. Thus, the apparatus and method are extensible without propagating a dependency up into the higher levels of the management software.

Also note page 7, lines 12 - 23:

> To solve these problems, a method and an apparatus are used to derive consistent service health measures by combining various instrumentation from both internal sources and external sources that relate to the service under observation. The service health metrics may be directly measured or derived from the applications, processes and thread instrumentation, for example. The method is independent of specific provider applications and management tool sets, thereby allowing for shorter time-to-market for service management solutions.
> The output of the method may be either in the form of a programmatic or scriptable interface to be used by high-level monitoring tools that are capable of reporting status of many disparate computer services. The tools may reside on different systems and architectures and may be supplied by different vendors. To accommodate different performance monitoring tools, the interfaces are <u>generic</u> and flexible.

Finally, note page 13, lines 15 - 17:

> The rules set 127 provides algorithms and rules to translate the metrics supplied by the data collection engine 121 into a generic format that is usable by the performance monitoring tools.

As the above-quoted passages from the specification demonstrate, the term "generic output" is an interface that allows the health metrics to be used without any need to know how the health metrics were derived, and that removes any dependencies between the services, their implementation, and the management tool set.

Leymann and claim 1 both use the word 'generic." But, any person skilled in the art will appreciate that the use of generic in claim 1 and the use of generic in Leymann are for very different purposes. Again, Leymann discloses a generic measuring agent, and Leymann's generic measuring agent need be modified only once. The invention of claim 1 recites a generic output.

Because Leymann and claim 1 use completely different generic features, Leymann cannot be used in combination with ARM API to render the subject matter of claim 1 obvious. Furthermore, as noted above, combining Leymann and ARM API will result in a non-functional device.

For all the reasons discussed above, claim 1 is patentable over ARM API in view of Leymann.

Claims 11, 18, and 21

Independent claims 11, 18, and 21 all recite features similar to those in claim 1 that are not disclosed or suggested by ARM API and Leymann, individually and in combination. Accordingly, claims 11, 18, and 21 also are patentable.

Dependent Claims

Claims 2 and 4 - 8 depend form patentable claim 1; claims 12 and 14 - 17 depend from patentable claim 11; claims 19 and 20 depend from patentable claim 18; and claims 22 and 23 depend from patentable claim 21. For these reasons and the additional features they recite, claims 2, 4 - 8, 12, 14 - 17, 19, 20, 22, and 24 also are patentable.

In view of the above amendments and remarks, Applicants respectfully submit that the application is in condition for allowance and request withdrawal of the rejection of the claims under 35 U.S.C. § 103(a).

A two-month extension fee is believed to be due. Please charge the fee to our Deposit Account No. 50-2849. Should any other fees be needed to prevent abandonment of this application, the Examiner is hereby authorized to charge the fees to our Deposit Account No. 50-2849.

Should the Examiner believe that anything further is desired in order to place the application in even better condition for allowance, the Examiner is invited to contact Applicants' undersigned representative at the telephone number listed below.

Respectfully submitted,

[illegible]

Date: December 22, 2008

John K. Harrop
Registration No. 41,817
Andrews Kurth LLP
1350 I Street, NW, Suite 1100
Washington, DC 20005
Tel. (202) 662-2700
Fax (202) 662-2739